# Introduction to Data Science — Assignment 3

Instructors: **Dr. Bahrak, Dr. Yaghoobzadeh**

TA(s): Mohammad Taha Fakharian

Deadline: Tuesday, Farvardin 28th, 11:59 PM

## Introduction

In this assignment, we are going to work with PySpark, which is the Python API for Apache Spark. It enables you to perform **real-time**, **large-scale data processing** in a **distributed environment** using Python.
For the first step, we'll learn how to set up PySpark and its requirements, then we'll do some warm-up exercises, in order to learn how to use it. Finally, we'll use this knowledge to do some investigation on a dataset.

## Installation

We recommend you install and use PySpark on a linux environment, since it's more convenient and user-friendly. Java installation is one of the mandatory things in installing Spark. Try the following command to verify the Java version:

```
java -version
```

In case you do not have Java installed on your system, then install Java before proceeding to the next step. We recommend you use openjdk for a more convenient installation.
After that, you need to install Spark; however, as of v2.2, installing PySpark will install Spark too. To install PySpark, you can use this link.

## Warm-Up!

Now that we've installed PySpark successfully, let's do some warm-up exercises! Before following the steps, there are some tips when using PySpark:

- PySpark works with dataframes, so results of most of the operations-if not all- are dataframes. Also, since PySpark works lazily (you need to explain the definition later in the questions), you won't see the results of operations unless you show them or save them in a place. As a result, it's recommended to define a utility function, which inputs a dataframe and shows the number of rows and the first rows of that dataframe. This can help you in debugging.

- For better visualization of dataframes in jupyter notebooks, since they are rendered to html, you can add this block of code at the beginning of your notebook:

```
from IPython.display import display, HTML
display(HTML('<style>pre { white-space: pre !important; }</style>'))
```

- Throughout this assignment, you're allowed to do the steps any way you like using PySpark-from pure SQL to specific functions in the library- but don't convert data frames into Pandas data frames and continue the procedure! This is not the goal of this assignment. However, we recommend you use functions, as they make code cleaner, more user-friendly and more optimized. You can find out more using this link and this link. Good luck!

The provided dataset (stocks.csv) contains information about a stock market, with opening, closing, the highest and the lowest price with its volume for each day. Follow these steps:

1. Read the csv file correctly!
2. Find out about the schema of data.
3. For those records with closing price less than 500, select opening, closing and volume and show them.
4. Find out records with opening price more than 200 and closing price less than 200.
5. Extract the year from the date and save it in a new column.
6. Now, for each year, show the minimum volumes traded, shown in a column named 'minVolume'.
7. Follow quite the same procedure as the previous step, but now for each year and month, show the highest low price, shown in a column named 'maxLow'.
8. For the last step, calculate mean and standard deviation of high price over the whole data frame and show them in two decimal places.

## Main Task

Now that you've learnt the basics of PySpark, let's do some investigation on a real dataset! The provided dataset(spotify.parquet) contains information about songs streamed on Spotify, which is an audio streaming and media service provider. You have access to a song's album, artist, its musical characteristics and its release date.

Suppose you're recruited as a data scientist for Spotify! Using your knowledge in PySpark and also your ability to search in its documentation 🙂, you need to extract useful statistics from this dataset, so that it could satisfy the employer. There are some tips for your important task:

1. Check the schema at the very first step!
2. You may need to preprocess some columns, e.g. 'release_date', so that they're ready for the investigation.

3. Use aggregation, filtering and transforming functions for useful statistics!
4. For dealing with array columns, you may select specific elements from them or explode them, so that aggregation functions work.
5. Also, checking the Top-K records might be interesting!

## Questions

1. Read about how Spark and Hadoop work. What does the term 'lazy evaluation' mean for them? Explain with a simple example.
2. Your main task's dataset has about 1,200,000 rows, which makes it quite hard, and even sometimes impossible, to work with. Explain how parquet files try to solve this problem, compared to normal file formats like csv.
3. As you might have noticed, Spark doesn't save checkpoints. How can we enforce it to do so? This can help us if we have multiple computation steps and we don't want to wait a lot for the result.
4. Top companies stream their data on a regular routine, e.g. daily. How can we save data, so that we could filter it based on specific columns, e.g. date, faster than regular filtering?
5. Let's face off Pandas and PySpark in the data analysis arena! When does each library truly shine, and why? Consider factors like data size, processing complexity, and user experience.

## Notes

- Upload your work as a zip file in this format on the website: DS_CA3_[Std number].zip. If the project is done in a group, include all of the group members' student numbers in the name.
- If the project is done in a group, only one member must upload the work.
- We will run your code during the project delivery, so make sure your results are reproducible.